Parallel Pseudospectral Electronic Structure: II. Localized Møller–Plesset Calculations

MICHAEL D. BEACHY,[1] DAVID CHASMAN,[1] RICHARD A. FRIESNER,[1] ROBERT B. MURPHY[2]

[1] *Department of Chemistry, and Center for Biomolecular Simulation, Columbia University, New York, New York 10027*

[2] *Schrödinger, Inc., Portland, Oregon*

Received 23 December 1997; accepted 10 February 1998

ABSTRACT: We have developed a parallel version of our pseudospectral localized Møller–Plesset electronic structure code. We present timings for molecules up to 1010 basis functions and parallel speedup for molecules in the range of 260–658 basis functions. We demonstrate that the code is scalable; that is, a larger number of nodes can be efficiently utilized as the size of the molecule increases. By taking advantage of the available distributed memory and disk space of a scalable parallel computer, the parallel code can calculate LMP2 energies of molecules too large to be done on workstations. © 1998 John Wiley & Sons, Inc. J Comput Chem 19: 1030–1038, 1998

Keywords: pseudospectral; parallel; localized Møller–Plesset, scalable

Correspondence to: R. A. Friesner

Contract/grant sponsor: National Science Foundation; contract/grant number: CHE9217368

Contract/grant sponsor: National Institutes of Health, Division of Research Resources; contract/grant number: P41RR 06892

Contract/grant sponsor: MetaCenter

This article contains Supplementary Material available from the authors upon request to via the Internet at ftp.wiley.com/public/journals/jcc/suppmat/19/1030 or http://journals.wiley.com/jcc/

CCC 0192-8651 / 98 / 091030-09

Introduction

Over the past 20 years, a great deal of experience has been gained in the application of correlated *ab initio* electronic structure techniques to small molecules in the gas phase. From these calculations, a very good picture of the accuracy of various methods for different chemical properties has emerged, at least for molecules composed primarily of first row atoms. For example, it is generally accepted that second-order Møller–Plesset perturbation theory (MP2), if used with a sufficiently large basis set, will yield accuracy of a few tenths of a kilocalorie per mole for relative energies of structures where a chemical bond is not made or broken, as with hydrogen bonding energies or conformational energy differences.[1,2] For other types of systems, such as those containing a transition metal, it is more difficult to specify precise error bounds. Nevertheless, a database of results has been assembled and performance has been quantified for restricted classes of compounds.

Despite these advances, it has only been in the past 5 years that correlated methods have started to be applied routinely to larger molecular structures (e.g., ref. 2). For *ab initio* techniques, the major difficulty is the scaling of the method with system size. As the least expensive conventional correlated approach, MP2 scales formally as $N_{occ}N_{basis}^4$, where N_{basis} is the basis set size and N_{occ} the number of occupied orbitals. (Quantities proportional to the molecule size will be represented by subscripted N variables.) The scaling of canonical MP2 is therefore proportional to N^5, where N is some measure of molecule size. While this scaling can, in theory, be reduced by the use of cutoffs, this has proven to be rather difficult in practice. This is particularly true if large basis sets such as the Dunning triple-zeta correlation consistent basis[3] are utilized. Other *ab initio* correlation techniques, such as CCSD(T), scale as N^7 with no cutoffs and hence are truly intractable for large molecules.

Due principally to the work of Becke,[4–7] density functional theory (DFT) techniques have made remarkable advances in accuracy for chemical systems, and are now the method of choice for many problems, such as bond energies of small molecules. However, there are serious questions concerning its precision at the smaller energy scales needed to evaluate conformational energies and hydrogen bonding energies[2] and with larger systems, due to the erroneous treatment of dispersion energy by current DFT functionals. Thus, it cannot yet be said that DFT can provide benchmark calculations for a wide range of important properties, despite its attractive scaling with system size, typically in the N^2–N^3 range.

During the past 5 years, we have pursued the development of *ab initio*-correlated methods based on two fundamental ideas: localization of occupied molecular orbitals, and use of pseudospectral (PS) numerical methods to directly evaluate two-electron integrals over these localized MOs. The work of Saebø and Pulay in the mid-1980s[8–10] demonstrated that localized orbital methods, such as local MP2, yield energetics differences that are, if anything, superior to canonical MP2 methods, as basis set superposition error is to a great extent eliminated.[11] However, substantial savings of computation time and reduction of scaling for large systems was not demonstrated in practice for a variety of technical reasons. The pseudospectral implementation of LMP2 overcomes these difficulties by eliminating the necessity of four index transforms. Scaling of $N^{2.5}$ with system size was demonstrated in our previous work[1] for the cc-pVTZ(-f) basis (the Dunning basis[3] without f functions on first row elements or d functions on hydrogen), as compared with the N^5 scaling observed for canonical MP2 in Gaussian-92. Furthermore, even for a relatively small molecule (piperidine, with 237 basis functions), a factor of greater than 10 reduction in CPU time was obtained. Conformational energy differences for 36 small molecules were calculated with the LMP2 program and compared with experiment, canonical MP2 calculations, and DFT results.[1] Accuracy obtained with LMP2 was qualitatively superior to DFT and equivalent to MP2 if the identical basis set was used. It has been demonstrated,[2] however, that MP2 must use a large basis set to obtain accurate results. This further emphasizes the importance of the efficient pseudospectral implementation of LMP2.

The pseudospectral methodology opened the possibility of routinely carrying out LMP2 calculations on large molecules, thus providing benchmark results for conformational energy differences. Such results would have a major impact in evaluating the applicability of DFT methods to these systems or investigating the performance of molecular modeling force fields. Up to now, it has

not been possible to rigorously test molecular modeling energetics on systems that are sufficiently large to investigate the question of transferability of parameters from small systems. The availability of a benchmark data base would provide force-field developers with a new approach to quality control.

With this objective in mind, we have developed a parallel version of our PS-LMP2 algorithm (pPS-LMP2). The current program runs on the IBM-SP2-scalable parallel supercomputer. The importance of a parallel implementation is not just in reduction of time to solution, although this is certainly relevant. To carry out very large LMP2 calculations efficiently, it is essential to store the $n_{local}^2 N_{occ}^2$ two-electron integrals on disk. (Here, n_{local} refers to the size of the localized virtual space. It is represented with a lower case n because this value depends only on the basis set, not on the molecule size.) This is a much smaller number of integrals than is required in four-index-transform algorithms that do not use "direct methods (where the number of two-electron integrals is $N_{basis}^2 N_{occ}^2$), and it grows only as N^2, but nevertheless can become larger than the capacity of a single disk. A suitable parallel algorithm allows the integrals to be distributed across the disks on each node, thus removing this limitation to the size of molecule that can be studied.

The present study provides a detailed description of the pPS-LMP2 algorithm, as well as timing results displaying the scaling of CPU time as a function of the number of processors. Our goal for scalability is to achieve, for a given molecule, 90% efficiency with a number of processors large enough to carry out the calculation in reasonable wall-clock time, on the order of 6–12 total hours. As the problem size increases, more processors will be needed to achieve this target. With a good parallel algorithm, however, the ratio of communication to computation will diminish and the 90% efficiency region will persist to a larger number of nodes. These goals are met in the current pPS-LMP2 implementation.

With the pPS-LMP2 methodology in hand, generation of benchmark data for chemically interesting systems is immediately possible. Our initial studies have focused on conformational energetics of peptides, specifically the alanine tetrapeptide. Comparison of results for various conformations of the alanine tetrapeptide with a wide range of molecular mechanics force fields are reported in ref. 12.

In addition, the current parallel implementation of PS-LMP2 energies should be easily extendable to a parallel implementation of PS-LMP2 gradients, although this has not been completed at this time. The LMP2 energy and gradient calculations are very similar in both algorithm and code implementation.

Pseudospectral Localized MP2

The equation of interest in second-order localized Møller–Plesset theory, as formulated by Pulay and Saebø[9] is that of the second order energy:

$$E^{(2)} = \sum_{i \geq j} \langle \mathbf{K}_{ij} \tilde{\mathbf{C}}_{ji} \rangle; \quad \tilde{\mathbf{C}}_{ji} = (1 + \delta_{ij})^{-1} (4\mathbf{C}_{ij} - 2\mathbf{C}_{ji}) \tag{1}$$

Here, the $\mathbf{C}_{ij}$ matrices are the coefficients of the first-order correction to the Hartreet–Fock reference:

$$\Psi^{(1)} = \sum_{i \geq j} \sum_{p,q} C_{ij}^{pq} \Psi_{ij}^{pq} \tag{2}$$

The $\mathbf{K}_{ij}$ matrices are two-electron integrals for localized occupied orbitals i, j and a localized subset of virtual orbitals p, q. Here, and elsewhere, we use i, j to refer to localized occupied orbitals and p, q to refer to virtual atomic orbitals that have been orthogonalized to the occupied subspace.

The $\mathbf{C}_{ij}$ are obtained by the solution of the second-order residual equation:

$$\mathbf{T}_{ij}^{(2)} = \mathbf{K}_{ij} + \mathbf{F}\mathbf{C}_{ij}\mathbf{S} + \mathbf{S}\mathbf{C}_{ij}\mathbf{F} - \mathbf{S}\sum_k [F_{ik}\mathbf{C}_{kj} + F_{kj}\mathbf{C}_{ik}]\mathbf{S} = 0. \tag{3}$$

Here, $\mathbf{F}$ is the external Fock matrix, $\mathbf{S}$ is the external overlap matrix, and F_{ij} is the internal Fock matrix element between localized orbitals i and j. The derivation of eq. (3) is presented elsewhere,[8, 13, 14] so we do not repeat it here.

In solving for $E^{(2)}$, the most time-consuming calculation of the pseudospectral implementation is in generating the group of $\mathbf{K}_{ij}$ matrices, which requires approximately 85% of the total CPU time. Nearly all of the remaining time goes to solving the residual equation. Once the $\mathbf{C}_{ij}$ coefficient matrices are obtained, the evaluation of the second-order energy takes a trivial amount of time. It is in the calculation of the two-electron integrals that

the pseudospectral method[15–17] gains its great advantage. The two-electron integral $\langle ij|pq\rangle$ is calculated by a sum over grid points as:

$$K_{ij}^{pq} = \sum_g Q_i(g) R_p(g) A_{jq}(g) \qquad (4)$$

Here, Q_i is the least squares fitting operator for molecular orbital i, R_p is the physical space representation of virtual orbital p. $A_{jq}(g)$ is the three-center, one-electron integral over molecular orbital j and virtual orbital q given by:

$$A_{jq}(g) = \sum_{\mu\nu} c_{\mu j} c_{\nu q} A_{\mu\nu}(g) = \sum_{\mu\nu} c_{\mu j} c_{\nu q} \int \frac{\chi_\mu(1)\chi_\nu(1)}{r_{1g}} d\mathbf{r}_1 \qquad (5)$$

The Greek subscripts denote contracted (AO) gaussian basis functions, which are distinct from virtual atomic orbitals p, q in that they are not orthogonal to the occupied subspace.

We again emphasize that the advantage gained by the pseudospectral method is in the avoidance of the four-index transform for the two-electron integrals.[1] The cost of this transformation from the atomic orbital basis set integrals $\langle \mu\nu|\rho\sigma\rangle$ to the localized and virtual orbital integrals $\langle ij|pq\rangle$ (and, therefore, the cost of analytic LMP2) scales as $N_{occ}N_{basis}^4$. However, by a direct multiplication on the physical space grid [eq. (4)], the pseudospectral method can assemble the $\mathbf{K}_{ij}$ in order $n_{virt}N_{occ}^2N_{grid}$ time (where N_{grid} is the number of grid points). As n_{virt} is constant within a given basis set and N_{occ} and N_{grid} are proportional to the size of the molecule, we end up with a computational cost that grows as N^3. This is the determining factor in the cost of the calculation. The most expensive cost of assembly for $Q_i(g)$, $R_p(g)$, and $A_{jq}(g)$ is that for $A_{jq}(g)$, which formally scales as $N_{basis}^2N_{grid}N_{occ}$ with a small prefactor. This scaling is reduced through distance-dependent grid cutoffs, however, and does not interfere with the observed $N^{2.75}$ scaling of K_{ij}^{pq} construction.[1]

Two elements key to the numerical accuracy of the pseudospectral LMP2 are detailed in previous publications.[1,18] The first is the use of analytical integrals for inexpensive one- and two-center two-electron integrals. This allows for the use of a much sparser grid. The second is the use of a length scales algorithm, which takes advantage of the eight fold permutational symmetry of the two-electron integral to assure that the fitting operator Q_i is built exclusively from short-range basis functions when R_p and A_{jq} are also built from short-range basis functions. We highlight these two parts of the algorithm not only because of their importance in the success of the serial program, but also because the parallelization of the LMP2 code would have been much more straightforward without them.

Parallelization of PS-LMP2

Previous work has been done in the parallelization of Pulay and Saebø's orbital invariant MP2 formulation by Bernholdt and Harrison,[14] although localized orbitals were not utilized. Bernholdt and Harrison were confronted with the same problem that faces all analytic implementations of MP2, however—that of the four-index transform. Much work has been devoted to the parallelization of this limiting portion of the algorithm.[19–23] The pseudospectral method allows the transform to be avoided. In principle, the parallelization of pseudospectral methods is easily accomplished by a division of the sum in eq. (4) over nodes followed by a global sum over the nodes.

Despite the great computational advantages of pseudospectral calculations, they can still outgrow the capabilities of a single workstation rather quickly. Although the computational bottleneck can be overcome simply by extending the time allocated to the solution of the problem at hand, there is no simple manner in which we can circumvent the problem posed by the finite storage resources associated with a given computer architecture. A scalable algorithm, if implemented on a properly scalable machine, offers a viable solution to this problem by simply making use of more nodes.

WORK AND STORAGE DISTRIBUTION

In addition to a decrease in the wall-clock time needed to obtain results, a parallel system, such as the IBM-SP2, offers the advantage of scalable amounts of memory and disk space. Each node on the SP2 has significant memory and local disk space. This is important for LMP2 because the storage requirements of local MP2 grow as N_{occ}^2. As a reference point, alanine tetrapeptide in the cc-pVTZ(-f) basis (658 basis functions) requires about 1.1 gigabytes of disk storage during the

iterative solution of eq. (3). Storage problems during this solution are exacerbated by the fact that we have implemented the accelerated convergence scheme described by Pulay and Saebø[9,10] where the extrapolated coefficient matrices are obtained from a linear combination of the current and previous matrices:

$$\mathbf{C}^n \leftarrow \alpha_0 \mathbf{C}^n + \alpha_1 \mathbf{C}^{n-1} + \alpha_2 \mathbf{C}^{n-2} \qquad (6)$$

Thus, in addition to storage required for single copies of the $\mathbf{K}_{ij}$, $\mathbf{T}_{ij}$, and $\mathbf{C}_{ij}$ matrices, the $\mathbf{T}_{ij}$ and $\mathbf{C}_{ij}$ matrices from the previous two iterations must be kept. This extra storage could be avoided, but only at the cost of much slower convergence.

For large calculations, the distribution of $\mathbf{K}_{ij}$ storage equally across all nodes is important. This is achieved by assigning each node some set of localized orbitals i. For the set of all i assigned to a node, it stores all $\mathbf{K}_{ij}$ with $j \le i$.

In addition to the N_{occ}^2 disk requirements, memory requirements for pPS-LMP2 grow as N^2 due to the $N_{basis} N_{occ} N_{grid}$ growth of $A_{jq}(g)$ and the N_{occ}^2 growth of the $\mathbf{K}_{ij}$. [This analysis is based on the fact that $A_{jq}(g)$ calculation is repeated for groups of grid points that are fixed in size.] As the size of the molecule grows, it becomes impossible to store $Q_i(g)$, $R_p(g)$, and $A_{jg}(g)$ in memory for all localized orbitals i and j. When this happens, the calculation of the exchange integrals $\mathbf{K}_{ij}$ must be broken into groups (called alpha groups because they are based on the first orbital index) based on the number of $A_{j\nu}(g)$ that can be stored in memory along with the Q and R operators and the summed K_{ij}^{pq}. This breakdown means that the calculation of the $A_{\mu\nu}(g)$ must be either repeated for each alpha group or stored on disk after a single index transform as $A_{j\nu}(g)$. At present, the $A_{\mu\nu}(g)$ calculation is repeated. For small molecules the cost is negligible, but for a 6-31G** alanine pentapeptide calculation (with 510 basis functions) the recalculation accounts for 8% of the total time of the calculation. As the size of the molecule increases the number of times the $A_{\mu\nu}(g)$ calculation will be repeated also increases, so it seems clear that disk storage of the $A_{j\nu}(g)$ must soon be implemented.

It is due to the breakdown of the calculation of the $\mathbf{K}_{ij}$ into alpha groups that the length scales algorithm adds difficulty to the parallel implementation of PS-LMP2. The length scales algorithm[1,18] formulates the integral $\langle ij|pq\rangle$ as:

$$\begin{aligned} K_{ij}^{pq} = \sum_g \Big\{ & \Big[Q_p^{(SR)}(g) R_i^{(S+L)}(g) \\ & + Q_i^{(SR)}(g) R_p^{(LR)}(g) \Big] A_{jq}^{(S+L)}(g) \\ & + \Big[Q_j^{(SR)}(g) R_q^{(S+L)}(g) + Q_q^{(SR)}(g) R_j^{(LR)}(g) \\ & + Q_q^{(LR)}(g) R_j^{(LR)}(g) \Big] A_{ip}^{(LR)}(g) \Big\} \end{aligned} \qquad (7)$$

Each integral $\langle ij|pq\rangle$ is calculated in part from a sum on grid points including a product of $A_{jq}(g)$ and in part from a sum on grid points including a product of $A_{ip}(g)$. When, due to a breakdown into alpha groups, a limited number of $A_{jq}(g)$ for localized orbitals j is available, it becomes impossible to calculate K_{ij}^{pq} for all i, j. It is therefore necessary to calculate length scales contributions to exchange integrals $\langle ij|pq\rangle$ for orbitals i not in the current alpha group. In the serial version of the code, these length scales pieces of the sum are simply added to the correct record of the file containing the running sum on grid points for all $\mathbf{K}_{ij}$ matrices. Due to the distributed storage of the $\mathbf{K}_{ij}$ in the parallel version, however, it can be that the final storing place for the length scales contribution to a given $\mathbf{K}_{ij}$ is not on the local node. Therefore, when alpha groups are spread over different groups nodes, the length scales integrals are stored on the local disk and a communication cost is incurred at the end of the pseudospectral integration routines. This cost has a limit of $N_{occ}^2 n_{virt}^2$ when every localized orbital is in its own alpha group and on its own node. In practice, the cost is less due to the fact that each node is home to more than one localized orbital. There is no dependence on number of processors in this communication cost because each length scales piece needs to be sent to only one other location.

Analytical Integrals

As described in the serial PS-LMP2 article,[1] the LMP2 code employs analytical corrections for one-center integrals and some two-center integrals. While the total number of analytical correction integrals initially grows as N_{atom}^2, the asymptotic growth is $O(N_{atom}\log(N_{atom}))$. This is due to the fact the only two-center integrals, which are treated analytically, are ones for which the two atoms are connected by a localized bond.

The amount of work required for the correction integrals scales as the number of integrals calcu-

lated. As such, there is no way to devise a scalable parallel algorithm for the distributed calculation of analytic corrections. The computation-to-communication ratio would be on the order of $N/N\log(P)$. Currently, each node calculates the one- and two-center AO analytical corrections needed for the $\mathbf{K}_{ij}$ matrices stored on its local disk. This results in the duplication of some integral evaluation, but is less expensive than sending the integrals to other nodes. Although some AO integral evaluation is duplicated, none of the work is repeated in the four-index transform for these integrals. As each node has a subset of local orbitals, i, the four-index transforms for the locally stored orbitals are completed independently on each node.

One method that could be used within the current parallel analytical corrections algorithm to reduce the duplication of work would be to choose the localized orbitals on each node from spatially contiguous groups. In this case, the only two-center integrals that would be duplicated would be the ones spanning more than one group. This method has not yet been implemented, as the amount of time spent in the analytical integral routines is small (5–10% for caffeine, depending on the basis set) and decreases with molecule size.

To account for load imbalance in the analytical corrections, they are calculated before the pseudospectral calculations. The dynamic allocation of work in the pseudospectral part of the code is then used to prevent nodes from sitting idle.

PSEUDOSPECTRAL INTEGRAL EVALUATION

Within each alpha group, parallelization of the pseudospectral integration [eq. (4)] is easily accomplished by the dynamic assignment of sets of grid points by a shared counter and the subsequent gathering of the resulting partial sums to the storage node.

Total work in the assembly of the $\mathbf{K}_{ij}$ scales as $N_{occ}^2 n_{virt}^2 N_{grid}$ and results in $N_{occ}^2 n_{virt}^2$ integrals. In the straightforward case with only one alpha group, the computation-to-communication ratio therefore reduces to $N_{grid}/\log(P)$, where P is the total number of processors.

For cases with multiple alpha groups, the simplest scenario is when all processors participate in the building of all alpha groups. For n_{groups} number of alpha groups of size $n_{groupsize}$ each, computation costs remain on the order of $n_{groups} n_{groupsize} N_{occ} n_{virt}^2 N_{grid}$ or $N_{occ}^2 n_{virt}^2 N_{grid}$ because $n_{groups} n_{groupsize} = N_{occ}$. The communication cost incurred to globally sum the grid-point sums is again $N_{occ}^2 n_{virt}^2 \log P$. To this, we must add the cost of communicating the length, scales pieces, which is, in its worst case, equal to $N_{occ}^2 n_{virt}^2$ (see ''Work and Storage Distribution'' subsection for analysis). Therefore, the total worst-case communication scenario is $N_{occ}^2 n_{virt}^2 \log(P) + N_{occ}^2 n_{virt}^2$, which, with the above computation cost, yields the final favorable computation-to-communication ratio of $N_{grid}/1 + \log(P)$. In practice, the communication cost is lower, as alpha groups are generally distributed on a subset of the total number of nodes in order to reduce the grid-point sum communication. (For example, in a molecule that needs three alpha groups, and is being run on four processors, nodes 0 and 1 would compute the first alpha group, 2 and 3 the second, and all four would work on the third for load balance to be achieved.) Also, the worst-case grid-point sum communication and worst-case alpha-group communication scenarios cannot exist simultaneously.

Before the analytic corrections can be added to the pseudospectral integrals, the pseudospectral terms corresponding to these corrections must be subtracted out. The calculation of these pseudospectral correction terms proceeds as with the standard pseudospectral integrals, across sets of grid points. Computational cost is again proportional to the number of integrals needed times the number of grid points, and communication costs are the same as in the above analysis.

SOLUTION OF RESIDUAL EQUATION

In each iteration of the solution of the residual equation [eq. (3)], there are $N_{occ}(N_{occ}+1)/2$ $\mathbf{T}_{ij}$ matrices to calculate. Each requires four matrix multiplies at a cost of n_{virt}^3 per multiply, a summation on k at cost $2N_{occ}n_{virt}^2$, and multiplication of the summed matrices at cost $4n_{vtrt}^2 N_{basis}$. The most expensive term in the total work cost reduces to $2N_{occ}^2 N_{basis} n_{virt}^2$, which again scales with problem size as N^3, as the number of virtuals remains constant within a given basis set. The even distribution of the $\mathbf{K}_{ij}$ across all nodes makes the communication of the $\mathbf{C}_{ij}$ necessary in the solution of eq. (3). Each node stores all $\mathbf{K}_{ij}$, $\mathbf{C}_{ij}$, and $\mathbf{T}_{ij}$ for its set of localized orbitals i with all $j \le i$. Updating of the $\mathbf{C}_{ij}$ in eq. (6) can proceed independently on each node, as each stores the corresponding parts of both $\mathbf{T}_{ij}$ and $\mathbf{C}_{ij}$. But, for the calculation of the $\mathbf{T}_{ij}$ [eq. (3)], we need $\mathbf{C}_{ik}$ and $\mathbf{C}_{ki}$ for all localized orbitals k. As each node stores only $\mathbf{C}_{ij}$ with $i \ge j$,

the $\mathbf{C}_{ij}$ with $i < j$ must be received in communication from other nodes. This communication cost reaches its limit in the instance where each node holds the $\mathbf{K}_{ij}$ for only one localized orbital i. In this case, the largest communication term is equal to $Pn_{virt}^2 N_{occ}^2/2$ per iteration. The computation-to-communication ratio for an iteration therefore reduces to the ratio of $4N_{basis}/P$, which is favorable as long as the number of basis functions is larger than the number of processes.

In addition to the communication of the $\mathbf{C}_{ij}$ matrices, a number of synchronization points are needed within the solution of eq. (3). The first is in the calculation of the energy [eq. (1)], for which each node contributes the trace of $\langle \mathbf{K}_{ij} \tilde{\mathbf{C}}_{ji} \rangle$ for the i, j resident on the local disk. The partial sum for each node must be collected into a global sum on every iteration to check for convergence of the second order perturbation energy. The second global sum needed is in the calculation of the coefficients for the conjugate-gradient-type accelerated convergence scheme. The communications cost in each of these cases is trivial, as the global sums are for less than ten floating point numbers each time.

Results and Discussion

We have run timings on caffeine in the 6-31G** and cc-pVTZ(-f) basis sets, the alanine tetrapeptide in the cc-pVTZ(-f) basis and the alanine penta- and decapeptides in the 6-31G** basis. Results for these test cases are presented in Figures 1 and 2. All timings were run on the joint Columbia Chemistry/Lamont Dougherty Earth Observatory IBM-SP2, using only 256M thin nodes.

As can be seen by comparing the speedups for caffeine, alanine pentapeptide, and alanine decapeptide (Fig. 1) an increase in the size of the molecule results in an increase in the number of nodes that can be used efficiently, thus demonstrating the scalability of the algorithm. For caffeine and alanine pentapeptide, an increase of basis functions by a factor of two results in a factor of more than two increase for the number of nodes that can be run on with the same efficiency. Assuming near 100% efficiency for the four-node case, alanine decapeptide shows a higher efficiency than alanine pentapeptide at 8 and 16 nodes. Scalability is also demonstrated in the cc-pVTZ(-f) results, with higher speedups being achieved for alanine tetrapeptide than caffeine when more than two nodes are used.

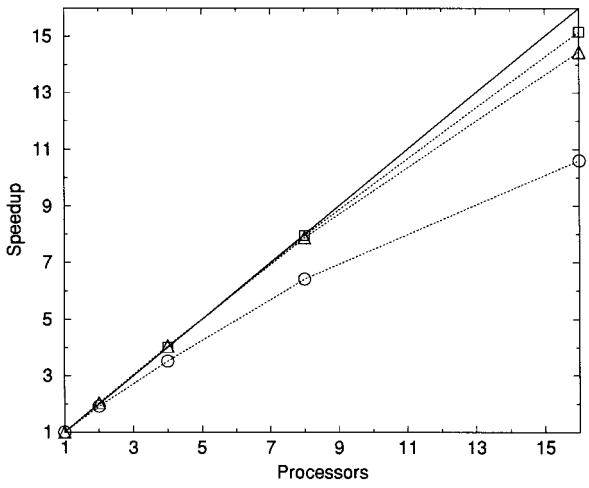

FIGURE 1. Parallel speedups for the 6-31G** test molecules: caffeine (○), with 260 basis functions; alanine pentapeptide (△), with 510 basis functions; and alanine decapeptide (□), with 1010 basis functions. Speedups for caffeine and alanine pentapeptide are shown in comparison to the one-node timings (2388 and 26,409 seconds, respectively). Speedups for alanine decapeptide are in comparison to the four node case (82,108 seconds).

We note a number of superscalar speedups: alanine pentapeptide in the 6-31G** basis for two and four nodes, and caffeine in the cc-pVTZ(-f) basis for two nodes. These result from the lower memory requirements in the distributed solution

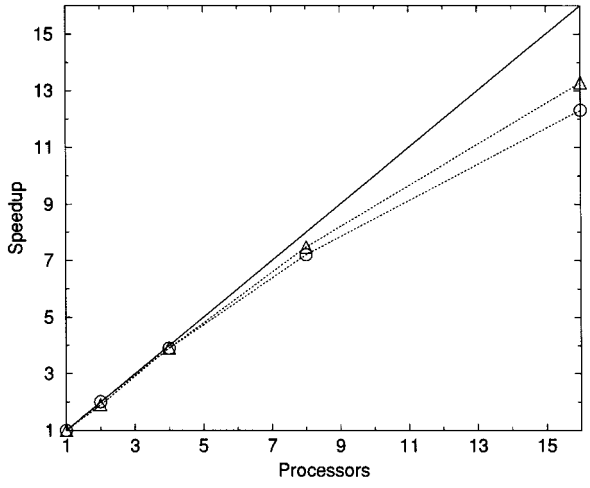

FIGURE 2. Parallel speedups for the cc-pVTZ(-f) test molecules: caffeine (○), with 412 basis functions; and alanine tetrapeptide (△), with 658 basis functions. Speedups are in comparison to the one-node timings (11,225 and 45,110 seconds, respectively).

of the residual equation. Due to the use of less memory per node, the file I/O proceeds at a much better real-time rate, as files are cached in memory. The better I/O performance also explains the superior two-node performance for caffeine in the cc-pVTZ(-f) basis compared with the two-node performance of the larger alanine tetrapeptide in the cc-pVTZ(-f) basis.

The cc-pVTZ(-f) basis (Fig. 2) calculations are somewhat less efficient than those for the 6-31G** basis. This is demonstrated by the fact that alanine tetrapeptide, which has 658 basis functions in the cc-pVTZ(-f) basis, achieves a lower efficiency than the alanine pentapeptide in the 6-31G** basis, which has only 510 basis functions. The lower parallel efficiency of the molecule with a larger basis set is due to the fact that there are more basis functions per atom, meaning that more of the poorly parallelized one- and two-center analytical integral corrections are calculated.

As in the case of the Hartree–Fock calculations described in part I of this study, we note that the parallel algorithm allows one to complete calculations on very large systems in a reasonable wall-clock time, while maintaining close to 90% computational efficiency. Taking 3 hours as the allotted time, we note that, in the 6-31G** basis, caffeine runs in 21 minutes on two nodes at 96% efficiency, whereas the alanine pentapeptide runs in 30 minutes on 16 nodes at 90% efficiency. In the cc-pVTZ(-f) basis, we see that caffeine runs in 26 minutes at 90% efficiency on eight nodes, whereas the alanine tetrapeptide runs in 1 hour and 41 minutes on eight nodes at 93% efficiency. The alanine decapeptide calculation has not yet been reduced to 3 hours. However, the computational efficiency on 16 nodes is at 95% and the calculation takes 6 hours and 1 minute. It is likely that increasing the number of nodes to 32 would yield a time near 3 hours and an efficiency of around 90%.

If we examine the four-node timings for the 6-31G** calculations, we see a demonstrated growth of $N_{basis}^{3.5}$, a factor higher than the value of 2.5 reported for the serial code.[1] The increase in this scaling factor is due to the repetition of work in calculating the $A_{\mu\nu}(g)$ multiple times. As mentioned previously, this will hopefully be overcome by saving the single-index-transformed $A_{i\nu}(g)$ to disk. The $N^{3.5}$ scaling of solution time with problem size is still close to the theoretically quoted N^3, and nearly two orders of magnitude better than that demonstrated by a canonical MP2 implementation.

Conclusion

The results presented herein demonstrate that pPS-LMP2 calculations can be carried out with large basis sets for molecules of quite respectable size in relatively short wall-clock times, without substantial sacrifice of parallel efficiency. For example, the alanine tetrapeptide in the Dunning cc-pVTZ(-f) basis requires less than 2 hours on eight nodes, and maintains an efficiency of greater than 90%. This is sufficiently fast turnaround that large numbers of computations can be carried out, which is the key to achieving impact on real-world chemical problems; where one is invariably interested in a series of molecules or multiple conformations. Calculations involving much larger molecules, with several thousand basis functions, will be reported in future publications.

The cc-pVTZ(-f) calculations carried out here for the alanine tetrapeptide, not even the largest case presented, would be very difficult to perform with any other *ab initio* electronic structure code. For example, if we utilize the results obtained in ref. 1 with Gaussian-92, a calculation involving 237 basis functions for the cc-pVTZ(-f) basis set required 1037 CPU minutes on an IBM 370. Assuming that the N^5 scaling reported for the cc-pVTZ(-f) basis set holds, the CPU time required for the 658-basis-function alanine tetrapeptide would be approximately 2850 CPU hours, or nearly 4 months. In actual fact, the calculation would undoubtedly consume much more time than this due to the vastly increased memory and disk requirements for such a large problem. It would be necessary to run in direct mode, numerous integral recalculations would be required, and the integral calculations for the cc-pVTZ(-f) basis would be quite demanding. Use of a supercomputer such as the Cray C90 would perhaps reduce the required time by an order of magnitude, but it would still be an extraordinarily expensive computation. However, evaluation of multiple conformational energies, as is required for benchmarking force fields, would again become prohibitive. One possible exception to our claim also highlights the great advantage of the pseudospectral LMP2 method. Wong et al.[19] demonstrated an MP2 time of 116.6 seconds for morphine (227 basis functions) on a 512-node Intel Touchstone Delta. Assuming N^5 scaling for their canonical MP2 algorithm, this translates into a time of 23,862 seconds for the 658-basis-function alanine tetrapeptide. This compares with the time

required to complete the alanine tetrapeptide (23,797 seconds) on two nodes of the SP2 using pPS-LMP2. For our algorithm, an additional increase in speed by a factor seven is easily achieved by increasing the number of nodes to 16. To do the same using the canonical algorithm, one would have to use a 3584-node Touchstone Delta, even with the assumption of perfect speedup.

Supplementary Material

The structure files for each of the molecules used in timing runs in this article are available in XMol's xyz format.

References

1. R. B. Murphy, M. D. Beachy, R. A. Friesner, and M. N. Ringnalda, *J. Chem. Phys.*, **103**, 1481 (1995).
2. A. St.-Amant, W. D. Cornell, P. A. Kollman, and T. A. Halgren, *J. Comput. Chem.*, **16**, 1483 (1995).
3. T. H. Dunning, *J. Chem. Phys.*, **90**, 1007 (1989).
4. A. D. Becke, *J. Chem. Phys.*, **96**, 2155 (1992).
5. A. D. Becke, *J. Chem. Phys.*, **97**, 9173 (1992).
6. A. D. Becke, *J. Chem. Phys.*, **98**, 5648 (1993).
7. A. D. Becke, *J. Chem. Phys.*, **98**, 1372 (1993).
8. P. Pulay, S. Saebø, and W. Meyer, *J. Chem. Phys.*, **81**, 1901 (1984).
9. S. Saebø and P. Pulay, *J. Chem. Phys.*, **86**, 914 (1987).
10. S. Saebø and P. Pulay, *J. Chem. Phys.*, **88**, 1884 (1988).
11. S. Saebø, W. Tong, and P. Pulay, *J. Chem. Phys.*, **98**, 2170 (1993).
12. M. D. Beachy, D. Chasman, R. B. Murphy, T. A. Halgren, and R. A. Friesner, *J. Am. Chem. Soc.*, **119**, 5908 (1997).
13. P. Pulay and S. Saebø, *Theor. Chim. Acta*, **69**, 357 (1986).
14. D. E. Bernholdt and R. J. Harrison, *J. Chem. Phys.*, **102**, 9582 (1995).
15. R. A. Friesner, *Chem. Phys. Lett.*, **116**, 39 (1985).
16. R. A. Friesner, *J. Chem. Phys.*, **85**, 1462 (1986).
17. R. A. Friesner, *J. Chem. Phys.*, **86**, 3522 (1987).
18. B. H. Greeley, T. V. Russo, D. T. Mainz, R. A. Friesner, J.-M. Langlois, W. A. Goddard III, R. E. Donnelly Jr., and M. N. Ringnalda, *J. Chem. Phys.* **101**, 4028 (1994).
19. A. Wong, R. Harrison, and A. Rendell, *Theor. Chim. Acta*, **93**, 317 (1996).
20. I. M. Nielsen and E. T. Seidl, *J. Comput. Chem.*, **16**, 1301 (1995).
21. L. A. Covick and K. M. Sando, *J. Comput. Chem.*, **17**, 992 (1996).
22. A. C. Limaye and S. R. Gadre, *J. Chem. Phys.*, **100**, 1303 (1994).
23. J. D. Watts and M. Dupuis, *J. Comput. Chem.*, **9**, 158 (1988).